Performance Modeling: Understanding the Present and Predicting the Future

David H. Bailey

Lawrence Berkeley National Laboratory

http://crd.lbl.gov/~dhbailey

Allan Snavely

University of California, San Diego

http://www.sdsc.edu/~allans

This presentation is available at:

http://crd.lbl.gov/~dhbailey/dhbtalks/dhb-perf-model.pdf

Performance Modeling

Objective: Gain an understanding of a computer system's performance profile, and then encapsulate this understanding in a compact formula.

Performance Profiles

Performance profiles depend on numerous factors, including:

- System size.
- System architecture.
- Processor speed.
- Multi-level cache latency and bandwidth.
- Interprocessor network latency and bandwidth.
- System software efficiency.
- Type of application.
- Algorithms used.
- Programming language used.
- Problem size.

Applications of Performance Modeling

- Runtime estimation.
  - Enable a scientist to predict how run time will change if job parameters are changed.
- System design.
  - Quantitatively decide between two competing technology options.
- System tuning.
  - Diagnose and rectify misconfigured channel buffers (Hoisie, 2000).
- Application tuning.
  - Determine how cache hit rates would change with various array blocking schemes.
- Algorithm choice.
  - Use bit-reverse FFT or stride-one-with-transpose FFT?
- System procurement.

LBNL's Seaborg System

- 6000-CPU IBM SP: 10 Tflop/s (10 trillion flops/sec).
- Currently #21 on Top500 list (a larger system is now being procured).
- Used for basic science projects funded by US Dept of Energy.

Large System Procurements

- At present, most laboratories utilize a set of application benchmarks, often adapted from user codes.
- Prospective vendors must devote highly expert staff to analyze and tune these codes for top performance.
- This process is very costly for vendors – a bid for a large system may cost a computer vendor over $1 million.
- These costs must then be recovered in the form of increased prices charged to successful system placements.

Can the procurement system selection process be streamlined?

If so, this will be a win for both laboratories and vendors.

Basic Methodology of Performance Modeling

- Summarize requirements of applications, using techniques that are manageably expensive, but still accurate.
- Obtain the application signatures automatically.
- Generalize the signatures to represent how application would stress arbitrary machines.
- Extrapolate the signatures to larger problem sizes that what can be actually run at the present time.

Analyzing Applications

- Statically analyze, then instrument and trace an application on some set of existing machines.
- Summarize, on-the-fly, the operations performed by the application.
- Tally operations indexed to the source code structures that generated them.
- Perform a merge operation on the summaries from each machine.

This yields information on memory access patterns (stride and range of memory accesses) and communications patterns (sizes and type of communication operations).

Application Signatures

- Conduct a series of experiments tracing a program.
- Analyze the trace by pattern detection to identify recurring sequences of messages and load/store operations.
- Ignore infrequent paths through the program, and drop sequences that map to insignificant performance contributions.

Example: In the NAS CG benchmark (over 1000 lines long), 99% of execution time is spent in this loop:

```
do k = rowstr(j), rowstr(j+1)-1
   sum = sum + a(k)*p(colidx(k))
enddo
```

Machine Signatures

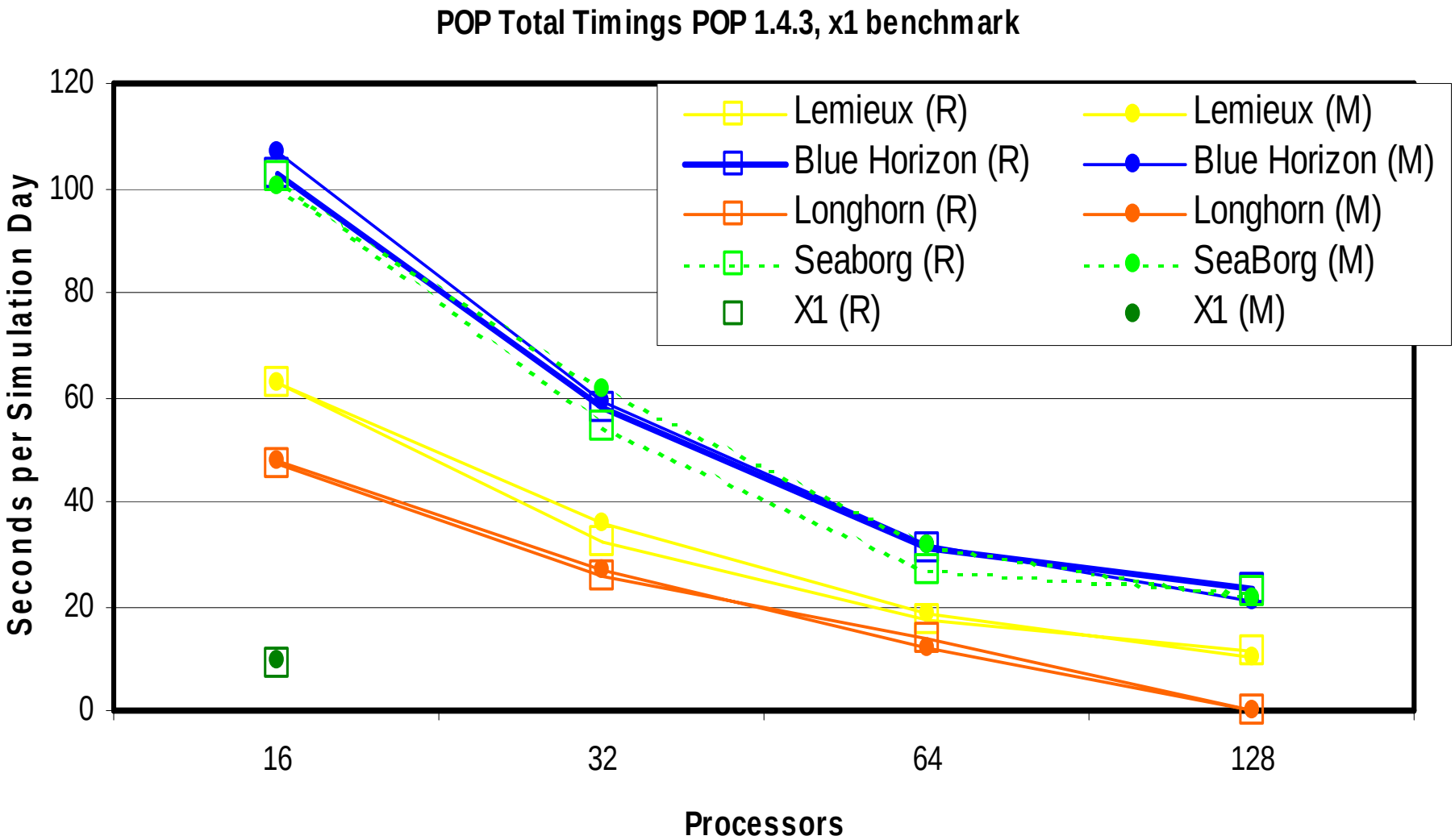
Use low-level benchmarks to gather machine profiles:

- High-level representations of the rates at which machines can carry out basic operations (such as memory loads, stores and message passing).
- Capabilities of memory units at each level of the memory hierarchy.
- Ability of machines to overlap memory operations with other kinds of operations (floating-point or communication operations).
- Extend profiles to account for reduction in capability due to sharing.

Combining Application Signatures with Machine Signatures

- Convolution methods for mapping application signatures to machine profiles.
- Techniques for modeling interactions between different memory access patterns within the same loop.
- Techniques for modeling the effect of competition between different applications (or task parallel programs) for shared resources.
- Techniques for defining “performance similarity” in a meaningful way.

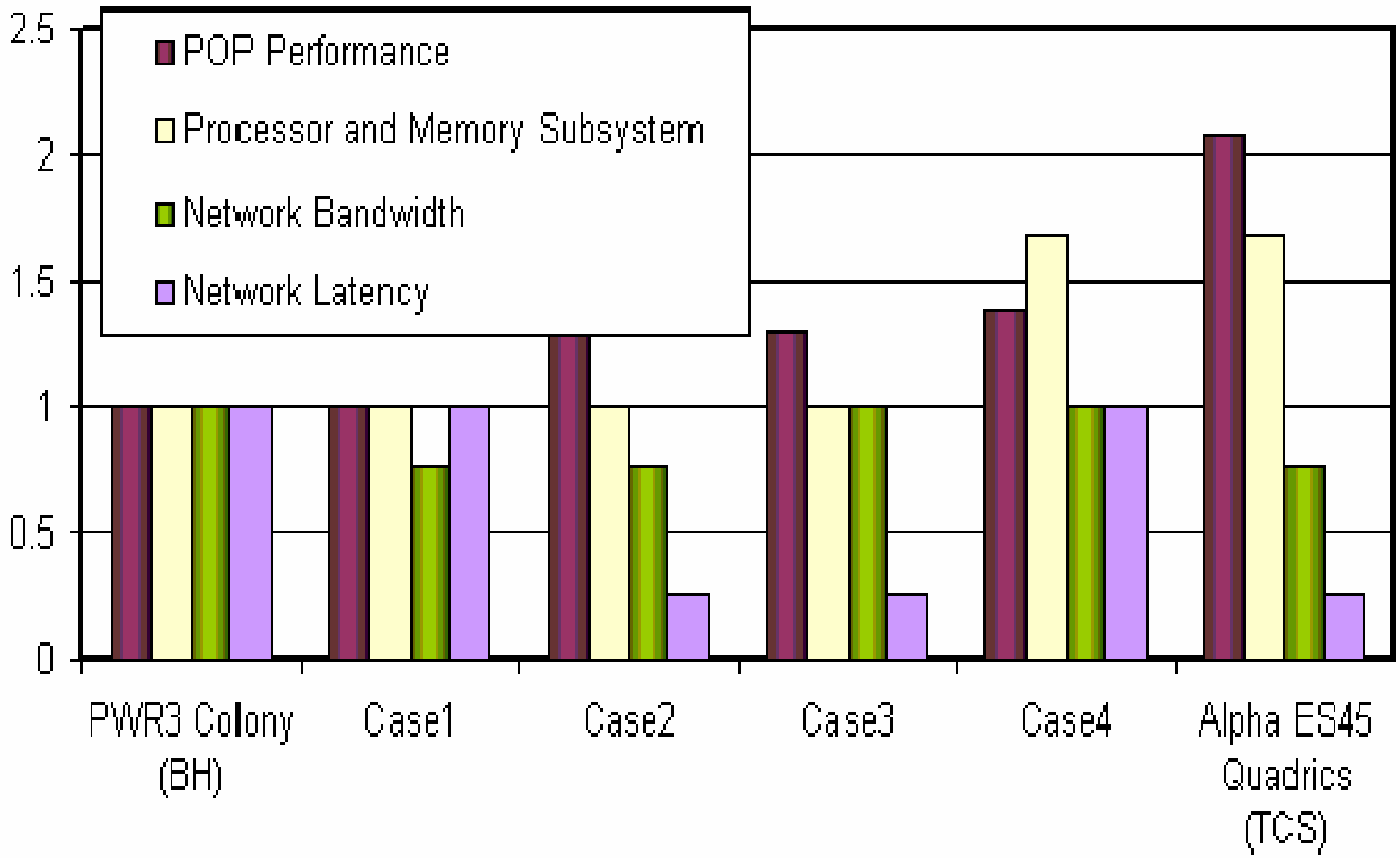
Results for Parallel Ocean Program (POP)



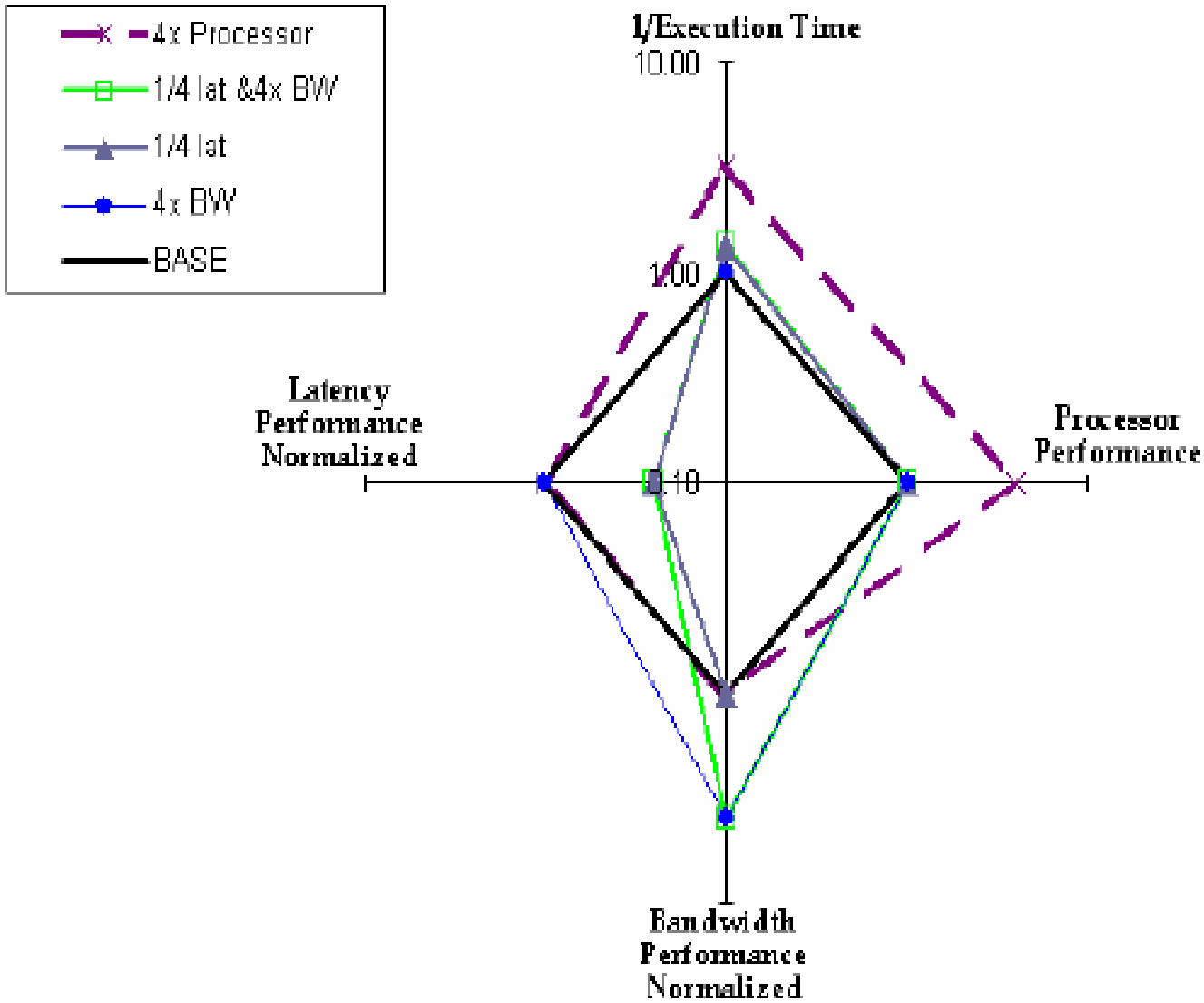
Accuracy of "Blind" Predictions on the US Department of Defense HPCMO Workload

Category	Ave. Absolute Error	Std. Deviation
Overall	**20.5%**	**18.2%**
AVUS std. input	15.0%	14.2%
AVUS large input	16.5%	16.2%
GAMESS std. input	45.1%	24.2%
HYCOM std. input	21.8%	16.7%
HYCOM large input	21.4%	16.9%
OOCORE std. input	32.1%	27.5%
Power3	17.4%	17.0%
Power4 p690	12.9%	9.6%
Power4 p655	15.7%	19.9%
Alpha	29%	17.6%
R16000	41.0%	18.5%
Xeon	28.2%	12.3%

“What If?” Performance Sensitivity Study



A Generalized Performance Sensitivity Study



Conclusion

Present:

- Performance models can be used for “what-if” analyses of changes to the application and/or computer system.
- Models can be used to help to system designers, helping them to optimize system parameters for certain applications.
- Models can be used to help computing centers select the best system in an acquisition.

Future:

- Modeling facilities can be embedded in user codes or compilers, thus enabling self-tuning scientific applications.

Challenges:

- Reduce effort and computer runs needed for accurate models.